Universidad de las Ciencias Informáticas

Ciudad de La Habana

Facultad 6

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN INFORMÁTICA

Título: Diseño e Implementación de un Manejador de Clúster para el Servidor de Streaming Distribuido ALLFRU´S.

AUTORA: Leidys López Cárdenas

TUTOR: Ing. Yoandrys S Pacheco Jerez

La Habana, julio del 2011

“Año 53 de la Revolución”

Por la ignorancia se desciende a la servidumbre, por la educación se asciende a la libertad.

Diego Luís Córdoba

DEDICATORIA.

A mi mamá por ser para mí la mejor madre, hermana y amiga.A la memoria de mi papá Alcides que aunque no está hoy, siempre me apoyó y creyó en mí.
A mi hermano Lázaro por estar siempre ahí para mí.

AGRADECIMIENTOS

A Dios por darme la vida y permitirme tener una familia tan maravillosa.
A la Revolución y a Fidel por haberme dado la oportunidad de estudiar en esta Universidad de Excelencia y hacer posible mi formación como ingeniera informática.
A mi tutor Yoandrys Pacheco por todo el apoyo y dedicación a este trabajo, por todos los señalamientos y recomendaciones, ha sido un honor trabajar con él.
A mis compañeros del Servidor de Streaming Distribuido Leo, Frank, Anyer, Yasiel y El Rene por haber sido mis compañeros de batalla en todo momento, por ayudarme a obtener las victorias. A Lisandra Sosa y su mamá Leticia por tratarme como a un miembro más de su familia.
A mis amigas AnaIvis, Vanessa, AnaBelkis, Roxana, Alianis, Mailín, Rocío, Diana, Yanet, Danae, Grechin por ser mis confidentes y amigas, y sobre todo por soportar mis malcriadeces.
A mí amigo Beny por ser mi Ángel de la guardia y por ser incondicional. A Adnan por ayudarme siempre que necesité de él. A mí amiga Lisandra Portillo por estar siempre ahí desde la primaria soportándome y por ser como mi hermana. A Misleidy por ser mi maestra y consejera. A mi hermano Lázaro por estar siempre ahí velando por mí. A mi tía Mima ser mi otra mamá, por cuidarme y quererme. A mis tíos Pimpo y Perly por ser tan especiales y por todas sus atenciones.
A mis primas Yarelys, Yerlanys y Yernay por estar siempre a mi lado y ser más que mis primas, por ser mis hermanas con sus virtudes y defectos, recuerden que las quiero. A mis tías Eva, Yoyi Pura por estar siempre al pendiente de mí. A mi abuela Eva y Zoila por todo su amor y cariño.
A mí bisabuela Regina y mi Abuelo Osvaldo por estar ahí cuando más los necesito.
A Nela por ser mi amiga, mi suegra y acogerme como si fuera su hija, por velar por mí y defenderme siempre aunque no tenga la razón,
A Zenaida por verme como su nieta, por cuidarme y ser mi cómplice.
A Dorney por ser mi amigo.

A mi novio Alejandro por su amor, comprensión, su espera y soportarme, por velar por mí.
A mí papá Alcides, aunque no estés más conmigo, por siempre creer en mí, por cumplir todos mis caprichos, por estar ahí para mí, por enseñarme el gusto por las matemáticas,

AGRADECIMIENTOS

A la persona más importante de mi vida, a la luz de mis ojos, al amor de mi vida, la única responsable de que yo esté aquí, por creer en mí, por ser mi mejor amiga, mi confidente, mi hermana, la persona que siempre me soporta todo, que me cuida, me entiende y me apoya de forma incondicional, que siempre no importa la distancia me ayuda, a esa persona la mejor madre del mundo le doy las gracias. A la mujer que algún día me gustaría llegar a ser. A ti Telly te agradezco por tu amor y confianza infinita.

DECLARACIÓN DE AUTORÍA.

Declaro que soy la única autora del presente trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2011.

_____	_____
Leidys López Cárdenas	Ing. Yoandrys S Pacheco Jerez
Autora	Tutor

RESUMEN.

La Universidad de la Ciencias Informáticas (UCI), fue creada con el objetivo de brindar soluciones de *software* que permitan automatizar la mayor cantidad de procesos en todos los sectores de la sociedad cubana. En la universidad varios de los proyectos en desarrollo utilizan para la transmisión de audiovisuales sobre redes de computadoras sistemas centralizados, debido a las dificultades presentadas por este tipo de servidores, el departamento de Señales Digitales de la facultad 6 decidió crear el proyecto DesComTec[1], orientado al desarrollo de un servidor *streaming* distribuido.

La presente investigación está encaminada al diseño e implementación de un componente Manejador de clúster con el objetivo de administrar las peticiones de servicio realizadas al servidor ALLFRY'S, permitiendo dirigir el flujo interno del mismo. Para dar cumplimiento al objetivo planteado el documento recoge los elementos conceptuales que fundamentan la investigación, así como el análisis de las diversas soluciones existentes en el mundo para brindar un servicio *Streaming.* También se describen las herramientas utilizadas en la implementación del componente. Se refleja el proceso de análisis, diseño, implementación y pruebas de la solución propuesta; presentando en cada flujo los resultados obtenidos que permiten validar la propuesta de solución desarrollada.

Palabras claves: *Streaming*, servidor *streaming* distribuidos, Manejador de Clúster.

[11] DesComTec(Desarrollo de Componentes y Tecnologías)

Índice de Figuras

Índice de Tablas

INTRODUCCIÓN

Desde sus inicios y trascendiendo en el tiempo los archivos de audio y video han sido utilizados tanto para la divulgación como para el entretenimiento. Inicialmente se utilizó tecnología analógica para su captura, transmisión, almacenamiento y reproducción. Gracias al desarrollo de las tecnologías de la información y las comunicaciones (TIC), su convergencia; expansión y desarrollo de Internet, surgen nuevas posibilidades de servicios sobre redes de computadoras. Debido a esto la industria del audiovisual ha tenido que adaptar sus contenidos a nuevas demandas, propiciando el surgimiento y desarrollo del audio y el video digital.

En un principio los métodos tradicionales para la entrega de video en Internet requerían, antes de su visualización, la descarga total del archivo de video en la computadora del usuario. En la mayoría de los casos este proceso era muy lento, dado el estrecho ancho de banda de las conexiones y la escasez de los recursos disponibles, haciéndose necesaria la creación de una nueva tecnología que suministrara a los usuarios la posibilidad de adquirir archivos audiovisuales en un menor tiempo de ejecución, sin necesidad de descargarlos.

En 1995 surge la tecnología *Streaming*, popularizada por la compañía *Real Networks*. Esta tecnología se basa en un sistema que permite acceder a un archivo audiovisual situado en un servidor sin proceder a su descarga y admite la distribución del archivo tanto en una intranet corporativa como en Internet. La tecnología *Streaming* tiene infinidad de aplicaciones, que aumentan a medida que la tecnología progresa directamente proporcionada por las demandas de los usuarios, predominando en el mundo de los negocios con las juntas virtuales y la entrega de noticias, en el entretenimiento con la transmisión de deportes, música y cine, en la educación con el aprendizaje a distancia y en la publicidad.

Para obtener cualquier servicio *streaming* se hace necesaria la presencia de un servidor especializado, clientes y un medio de comunicación. En el caso de los servidores es imprescindible la utilización de una arquitectura que soporte el proceso de transmisión de audiovisuales así como las disposiciones de los usuarios. Existen diferentes tipos de arquitecturas para estos propósitos como: la centralizada, la independiente, la jerárquica y la distribuida, cada una con sus ventajas y desventajas.

Actualmente en la Universidad de las Ciencias Informáticas (UCI) algunos de los proyectos en desarrollo emplean para la transmisión de audiovisuales servidores *streaming* centralizados. Este tipo de servidor ha demostrado ser poco tolerante a fallos e incapaz de manejar gran cantidad de conexiones. Debido a esto en la universidad se creó el proyecto de Desarrollo de Componentes y Tecnologías[2], enfocado a la confección del servidor *streaming* distribuido ALLFRY'S. El proyecto DesComTec para lograr mejores resultados en la creación del servidor fraccionó el funcionamiento de este en diferentes componentes, donde cada módulo cumple una tarea esencial en la cadena de transmisión de audiovisuales.

Por lo expuesto anteriormente el trabajo de investigación está encaminado a darle solución al siguiente **problema:** ¿Cómo gestionar el funcionamiento e interacción entre los módulos que componen el servidor *streaming* distribuido ALLFRY'S?

A partir del problema planteado el **objetivo general** de esta investigación está dirigido a: Desarrollar un Componente Manejador de Clúster para el servidor *streaming* distribuido ALLFRY'S, identificando como **objeto de estudio:** Las tecnologías libres para el desarrollo de un Manejador de Clúster para un servidor *streaming* distribuido, enfocando el **campo de acción** en: Las características y funcionalidades que proveen las tecnologías libres para el desarrollo de un Manejador de Clúster para servidores *streaming* distribuidos.

La validación de la investigación está sustentada en la siguiente **idea a defender:** La implementación de un Manejador de Clúster permitirá la administración de las peticiones de servicio realizadas al servidor ALLFRY´S, así como dirigir el flujo interno del servidor.

Para dar cumplimiento al objetivo general se trazaron los siguientes **objetivos específicos**:

- Caracterizar los procesos relacionados con el Manejador de Clúster para Servidores de *Streaming* distribuido.
- Desarrollar un componente de Manejador de Clúster con las funcionalidades requeridas por el servidor *streaming* distribuido del proyecto DesComTec.

Para dar cumplimiento a los objetivos específicos se definieron las siguientes **tareas de investigación**:

[2]**Desarrollo de Componentes y Tecnologías (DesComTec)**

- Identificación de los estándares nacionales e internacionales utilizados para el desarrollo de un componente Manejador de Clúster para un servidor *streaming* distribuido.
- La identificación de las funcionalidades del Manejador de Clúster para un servidor *streaming* distribuido.
- Desarrollo de la documentación teórica del tema de interés.
- Justificación de la Metodología de Desarrollo de *Software* a usar en el proceso.
- El diseño del componente Manejador de Clúster para un servidor *streaming* distribuido.
- La implementación del componente Manejador de Clúster para un servidor *streaming* distribuido.
- La gestión del proceso de pruebas al componente.

Se utilizaron como métodos de investigación científica:

Métodos teóricos:

Modelado: Permite una abstracción completa del objeto con la realidad y capta la esencia del problema que se desea automatizar. De este modo el método permite conocer con exactitud cuáles son las características del componente mediante el modelado de los procesos del negocio, utilizando para este fin diagramas y figuras.

Análisis-Histórico-Lógico: Posibilita hacer un estudio de los antecedentes que puede tener el componente que se desea automatizar. Con la ayuda de este método se logró realizar un análisis de otras soluciones existentes, también fue utilizado con el objetivo de definir lenguajes de programación, entre otras tecnologías asociadas al desarrollo del componente.

Analítico-Sintético: Facilita la descomposición del tema en múltiples relaciones y componentes, sintetizando las partes analizadas para arribar a las conclusiones.

Breve descripción de la estructura del trabajo de diploma:

Capítulo 1: Fundamentación teórica.

En este capítulo se explican los conceptos asociados al dominio del problema, los cuales servirán de base para el diseño del componente Manejador de Clúster. Además, se analizan las soluciones existentes, proporcionando soporte teórico para dar solución al problema de la investigación.

Capítulo 2: Tendencia y tecnologías actuales

En este capítulo se exponen las tecnologías, herramientas y metodologías utilizadas en el mundo para la construcción de aplicaciones de escritorio válidas en la construcción de la solución que se propone, dejando argumentada la decisión de usar RUP como metodología de desarrollo, UML como lenguaje de modelado, Visual Paradigm como herramienta CASE, C++ como lenguaje de programación y Qt Creator como entorno de desarrollo.

Capítulo 3: Presentación de la solución propuesta.

En este capítulo se confecciona el modelo del negocio en el que aparece enmarcado la propuesta del componente a desarrollar. Se definen los requerimientos funcionales y no funcionales del componente, se especifican los actores y casos de uso del sistema. Además, se expone la definición del modelo de análisis y el diseño del sistema.

Capítulo 4: Implementación y prueba de la solución propuesta.

En este capítulo se lleva a cabo la construcción del sistema, así como las pruebas del mismo. Se realiza el modelo de implementación que está compuesto por los diagramas de componentes y los diagramas de despliegue. También recoge el modelo de prueba.

Capítulo 1: Fundamentación Teórica.

1.1.Introducción

En este capítulo se exponen los elementos conceptuales que fundamentan la investigación de los Manejadores de Clúster para servidores *streaming*. Se presenta un análisis de algunas soluciones existentes que permiten la transmisión de audiovisuales sobre redes de computadoras. El estudio de estas soluciones permitirá dar respuesta a la siguiente interrogante: ¿Por qué las aplicaciones existentes no solucionan el problema de investigación?

1.2.Conceptos asociados al dominio del problema

- ***Streaming***

"*Streaming*" una de las palabras más populares en Internet, debido a que las redes sociales transmiten videos en *streaming*, las radios *online* hacen lo propio con el audio y en la actualidad incluso las películas se obtienen en esta modalidad. Se pueden mencionar diversos ejemplos que utilizan este término, pero ¿Qué se entiende por *streaming*?

Según J. Koldobika Espinosa "*se llama streaming a la capacidad de distribuir contenidos multimedia a través de una red digital, con la característica especial de permitir el acceso a estos contenidos según se requiera, sin necesidad de descargarlos previamente. Las aplicaciones basadas en streaming pueden clasificarse en aquellas relacionadas con la interacción entre dos o más usuarios (como videoconferencia y transmisión de voz IP), y aquellas que principalmente se dedican a distribuir contenidos multimedia generalmente a múltiples destinos, tanto en la modalidad de distribución en directo como en la modalidad de distribución bajo demanda."* (Espinosa, y otros, 2006)

Después de analizadas las diferentes fuentes bibliográficas la autora de la investigación define como *streaming* la tecnología que permite la recepción instantánea de las medias que fluyen desde un servidor. Por lo planteado anteriormente y después de un estudio realizado se puede decir que esta tecnología contiene un conjunto de productos y técnicas cuyo objetivo es la entrega de archivos multimedia, como audio y video, sobre redes de computadoras.

Distribución del *streaming*:

Unicast: Utiliza una conexión punto a punto en la que se establece comunicación entre dos elementos de forma bidireccional. Esta es la forma en la que funcionan la mayor parte de las conexiones en Internet y presenta la ventaja de ser soportada por todos los tipos de equipos. (Agüera, 2004)

Multicast: No emite para todos los destinos de una red, se trata de una conexión uno-a-varios, no uno-a-todos. El proceso de conexión implica el envío de un datagrama IP a un grupo de equipos identificados por una sola dirección IP. Esto es lo que se denomina un grupo multicast. Sólo aquellos equipos que han realizado una petición de que le sean enviados los datos estarán incluidos en las direcciones del datagrama, de forma que el resto de los equipos los ignorarán. Esto establece en realidad una red virtual en la que las direcciones multicast se asignan a grupos estables o dinámicos. (Agüera, 2004)

Broadcast: Define un tipo de conexión uno-a-todos, es decir, que la comunicación se realiza entre un nodo emisor y los puntos que le rodean. Los datagramas IP enviados durante este tipo de conexiones incluyen una dirección de destino correspondiente a todos los equipos conectados a la misma red local. Es un tipo de conexión utilizada en intranets fundamentalmente para la difusión programada de contenidos, aunque no es especialmente eficiente en la administración de recursos. (Agüera, 2004)

- **Servidor**

Término que proviene del latín *servitor* y cuyo uso ha cambiado en los últimos años. En la actualidad la noción de servidor está asociada al campo de la tecnología. En informática un servidor es una computadora que forma parte de una red y que provee servicios a otras computadoras, que reciben el nombre de clientes. Los servidores se suelen utilizar para almacenar archivos digitales. El cliente se conecta a través de la red de computadoras con el servidor y accede a los archivos en cuestión. En ocasiones la computadora puede cumplir con las funciones de servidor y de cliente de manera simultánea.

- **Servidor *Streaming***

Se entiende por servidor el conjunto de *hardware* y *software* necesario para cumplir la función principal de servir vídeo y proporcionar un flujo continuo de información (vídeo) en vivo o bajo demanda. Independientemente del tipo de arquitectura. (Lacort, 2007)

Los servidores *streaming* proporcionan dos tipos de contenidos:

VoD (*Video on Demand*), petición por clientes individuales de ficheros almacenados en el servidor, sobre los que tiene un control similar a un video doméstico (posicionamiento, parada, retroceso o avance rápido). (Méndez, 2010)

Difusión (*broadcast*) a varios clientes de un mismo contenido, ya sea creado en ese momento en vivo (*Live Broadcast*), o almacenado previamente en el servidor. (Méndez, 2010)

Se puede decir que a nivel técnico un servidor *streaming* es una aplicación que espera, procesa y sirve peticiones de uno o varios clientes. Para lograr este proceso el servidor necesita de una arquitectura que le permita atender las disposiciones de los clientes en la red.

- **Arquitectura *Software***

La representación de alto nivel de la estructura de un sistema o aplicación, que describe las partes que la integran, las interacciones entre ellas, los patrones que supervisan su composición, y las restricciones a la hora de aplicar esos patrones. (Fuentes, 2001) Para profundizar sobre la arquitectura del servidor streaming distribuido ALLFRY'S consultar el documento "Arquitectura del servidor de streaming distribuido ALLFRY´S"

Entre las arquitecturas de servidores *streaming* más populares se encuentran:

Centralizada: cuenta con un servidor centralizado al cual están conectados todos los usuarios del sistema, a través de una red de comunicación. En esta arquitectura la gestión de los usuarios se basa en un único nodo de servicio que centraliza la atención de todas las peticiones.

Independientes: en estos sistemas los usuarios están agrupados en segmentos de red local cuyo tráfico es independiente entre sí. El éxito de su funcionamiento radica en que las peticiones se pueden servir localmente sin necesidad de acceder a un servidor centralizado. Esto se logra colocando un servidor por cada red local y replicando el contenido.

Proxy: similar a la arquitectura independiente, debido a que la red de los usuarios se segmenta en varias redes conectadas a servidores *proxy*. Su diferencia radica en que en el servidor proxy solo almacena los contenidos más populares, por lo que si llega una petición que no puede ser atendida debido a que el contenido no se encuentra en el servidor, la solicitud es remitida al servidor principal.

Distribuidos: en estos sistemas los distintos nodos de servicio tienen que colaborar entre sí para poder atender a los usuarios. La gestión de las peticiones y los contenidos multimedia se distribuyen entre los componentes del sistema, donde un cliente puede actuar de cliente y servidor a la vez, es decir, un cliente actúa como servidor para otro cliente.

- **Arquitectura de Servidor Multimedia Distribuida Adaptable (ADMS)**

Arquitectura que divide el funcionamiento del servidor *streaming* en cuatro componentes, que desempeñan una tarea esencial en el proceso de trasmisión de media. Entre los componentes que describe esta arquitectura se puede encontrar los distribuidores de datos (DD), el manejador de datos (DM), el colector de datos (CD) y el manejador de clúster (MC*).* (Tusch, 2003)

A continuación se explica el funcionamiento de cada uno de estos componentes dentro de la arquitectura ADMS:

Manejador de clúster (MC)

Es el componente central en la arquitectura ADMS. Es nombrado director del clúster, ya que gestiona la ubicación de los distribuidores datos y los colectores de datos, que resulta en un clúster de servidores de aplicaciones dinámicas ADMS. Por otra parte, representa el punto central para el manejo de las conexiones de los clientes, aunque no atiende las solicitudes del cliente por sí mismo, las dirige adecuadamente a un distribuidor o colector. (Tusch, 2003)

Adaptador de Núcleo (AN)

El adaptador de núcleo es un subcomponente del Manejador de Clúster. Es el encargado mediante varios algoritmos de selección, encontrar el candidato adecuado entre un conjunto dado de nodos. Este

componente no se encuentra dentro de la descripción original de la arquitectura. Este subcomponente es un aporte realizado por el proyecto DesComTec a la aplicación de la arquitectura ADMS en el servidor *streaming* distribuido ALLFRY'S. El Adaptador de Núcleo es fruto de disminuir las funcionalidades del Manejador de Clúster descrito originalmente por la arquitectura ADMS.

Distribuidor de Datos (DD)

Componente que distribuye los datos de medios recibidos desde un cliente de producción o una cámara fotográfica, se encarga de seccionar la media en paquetes y distribuirlos para su posterior almacenamiento. (Tusch, 2003)

Colector de Datos (CD)

El colector de datos es un componente del servidor *streaming* distribuido que realiza la operación inversa al distribuidor de datos. Este recolecta las unidades de bandas de un paquete de medias con la configuración apropiada del manejador de datos, recoge estos paquetes de diferentes orígenes de datos, después envía las unidades en secuencia de buffer hacia los clientes a través de una conexión. Este componente evita la sobrecarga en los servidores de almacenamiento. (Tusch, 2003)

Manejador de Datos (MD)

Un Manejador de Datos es el que provee de manera eficiente el almacenamiento y la recuperación de los datos en un servidor *streaming*, después que el componente Distribuidor de Datos realice la operación de separación de segmentos el Manejador de Datos es el encargado de almacenarlo. A la hora de recuperar dichos segmentos el componente Colector de Datos realiza una petición al manejador, el cual va a buscar donde lo tiene almacenado mediante algún parámetro definido por este. (Tusch, 2003)

- **Sistemas distribuidos**

Conjunto de computadoras o nodos, conectadas mediante una red, que se comunican mediante el paso de mensajes y cooperan entre sí, es decir, comparten recursos, brindan un servicio o ejecutan una aplicación para alcanzar un objetivo común. (Fuentes, 2001) Un ejemplo de sistemas distribuido son los cluster.

- **Protocolos**

Dado que los servidores *streaming* requieren la transferencia entre nodos de grandes volúmenes de contenido multimedia a una gran velocidad de ejecución, es necesario la utilización de protocolos de comunicación. Se entiende por protocolos de comunicación el conjunto de estándares y reglas que definen como se comunican entre sí dos máquinas, aplicaciones, o dos objetos en la Programación Orientada Objeto. La función principal de los protocolos de comunicación es la de tener un formato común para la realización de comunicaciones entre programas elaborados por personas completamente diferentes, es decir, es un conjunto de reglas y procedimientos que deben respetarse para el envío y la recepción de datos a través de una red. Existen diversos protocolos de acuerdo a cómo se espera que sea la comunicación.

Generalmente los protocolos se clasifican en dos categorías según el nivel de control de datos requerido:

Protocolos orientados a conexión("pesados"): estos protocolos controlan la transmisión de datos durante una comunicación establecida entre dos máquinas. En este caso, el equipo receptor envía acuses de recepción durante la comunicación, por lo cual el equipo remitente es responsable de la validez de los datos que está enviando. Los datos se envían entonces como flujo de datos. TCP[3]es un protocolo orientado a conexión.

Protocolos no orientados a conexión("ligeros"): éste es un método de comunicación en el cual el equipo remitente envía datos sin avisarle al equipo receptor, y éste recibe los datos sin enviar una notificación de recepción al remitente. Los datos se envían entonces como bloques (datagramas). UDP[4]es un protocolo no orientado a conexión.

Protocolos utilizados en la transmisión de *streaming*:

Protocolo de transporte en tiempo real (RTP)

RTP es un protocolo que proporciona soporte para el transporte de datos en tiempo real (flujos de vídeo y de audio) formado por RTP y Protocolo de control en tiempo real (RTCP). Este último es un protocolo de

[3] **TCP: Transmission Control Protocol**
[4] **UDP: User Datagram Protocol**

comunicación que proporciona información de control que está asociado con un flujo de datos para una aplicación multimedia (flujo RTP). Trabaja junto con RTP en el transporte y empaquetado de datos multimedia, pero no transporta ningún dato por sí mismo. Se usa habitualmente para transmitir paquetes de control a los participantes de una sesión multimedia de *streaming*. La función principal de RTCP es informar de la calidad de servicio proporcionada por RTP.

Recoge estadísticas de la conexión y también información como por ejemplo *bytes* enviados, paquetes enviados, paquetes perdidos entre otros. Una aplicación puede usar esta información para incrementar la calidad de servicio, ya sea limitando el flujo o usando un códec de compresión más baja. En resumen RTCP se usa para informar de la calidad del servicio.

Por otra parte, el protocolo RTP se creó específicamente para la transmisión de audio y vídeo, gracias a que incluye en su cabecera informaciones que sincronizan imagen y sonido, al tiempo que es capaz de determinar si se han perdido paquetes y si éstos han llegado en el orden correcto. Trabaja sobre UDP, tiene características especiales para el trabajo con sistemas en tiempo real. En un principio fue diseñado para emisiones multicast de tráfico en tiempo real (aunque también se puede utilizar en emisiones unicast) y puede ser utilizado para el video bajo demanda y servicios interactivos. En su contra, tiene que no asegura ni la entrega continua de información, ni la de todos los paquetes y no puede evitar la entrega desordenada de los mismos, aunque sí los controla. (Torres, 2009)

Protocolo de datagramas de usuario (UDP)

UDP es un protocolo del nivel de transporte basado en el intercambio de datagramas que proporciona una sencilla interfaz entre la capa de red y la capa de aplicación. No tiene confirmación, ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros; y tampoco se sabe si ha llegado correctamente, ya que no hay confirmación de entrega o de recepción.

Este protocolo se utiliza principalmente cuando el orden en que se reciben los mismos no es un factor fundamental, o también cuando se quiere enviar información de poco tamaño que cabe en un único datagrama. Es un protocolo no orientado a conexión. Cuando una máquinas A envía paquetes a una máquina B, el flujo es unidireccional.

La transferencia de datos es realizada sin haber ejecutado previamente una conexión con la máquina de destino (máquinas B), y el destinatario recibirá los datos sin enviar una confirmación al emisor (la máquinas A). Esto es debido a que la encapsulación de datos enviada por el protocolo UDP no permite transmitir la información relacionada al emisor. Por ello el destinatario no conocerá al emisor de los datos excepto su IP. No introduce ningún retardo para establecer una conexión, no añade ninguna mejora en la calidad de la transferencia; aunque si incorpora los puertos origen y destino en su formato de mensaje, no mantiene estado de conexión alguno y no realiza seguimiento de estos parámetros. Así, un servidor dedicado a una aplicación particular puede soportar más clientes activos cuando la aplicación corre sobre UDP. Es utilizado por aplicaciones como RCP (comando para copiar ficheros entre ordenadores remotos), pero sobre todo se emplea en tareas de control y en la transmisión de audio y video a través de una red ya que no hay tiempo para enviar de nuevo paquetes perdidos cuando se está escuchando a alguien o viendo un vídeo en tiempo real. (Torres, 2009)

Protocolo de *streaming* en tiempo real (RTSP).

RTSP es un protocolo no orientado a conexión y si orientado a corrección, en lugar de esto el servidor mantiene una sesión asociada a un identificador, en la mayoría de los casos RTSP usa TCP para datos de control del reproductor y UDP para los datos de audio y vídeo aunque también puede usar TCP en caso de que sea necesario. Este protocolo de presentación multimedia cliente / servidor permite controlar el flujo de datos multimedia sobre la red IP. Proporciona una funcionalidad de control remoto para audio y video como detener, ir hacia delante, atrás y posición absoluta entre el servidor y el cliente. Además es un protocolo de nivel de aplicación diseñado para funcionar con protocolos de bajo nivel como RTP, funciona tanto para grandes audiencias *multicast* como para *unicast*. Este resulta bastante eficaz debido a que permite una comunicación bidireccional de alta calidad, aporta un cierto nivel de seguridad y, lo mejor, es fácilmente adaptable a distintas plataformas como Windows, Unix o Macintosh. Para hacer llegar a cada usuario la información, hace falta, aparte del propio protocolo de transmisión que es el que indica cómo se enviará la información a nivel lógico por la red, un formato determinado que aplique o no compresión y que será el que almacenará la información en un fichero. Un protocolo que opera junto a RTSP es: RTP (El Protocolo de Transporte de tiempo Real). (Torres, 2009)

Protocolo de Control de Transmisión (TCP).

Protocolo orientado a conexión, localizado sobre la capa IP (Internet *Protocol*): Es un protocolo confiable, los datos siempre llegarán de forma ordenada, ya que implementa mecanismos de retransmisión en caso de que el receptor no confirme la recepción de los datos. También implementa mecanismos de control de congestión y control de flujo. Lo que implica, que TCP es más lento que UDP. Normalmente TCP se utiliza para mensajes de control, y no se recomienda para la transmisión de vídeo aunque a pesar de ello se utiliza en algunas aplicaciones de vídeo. TCP no permite envío de mensajes multicast o broadcast. (Torres, 2009)

Protocolo de Transferencia de Archivos (FTP).

El protocolo de transferencia de ficheros se encuentra sobre el protocolo TCP, al igual que el HTTP, permite obtener cualquier bloque de un fichero alojado en el servidor FTP. Algunos clientes, permiten reproducir directamente un vídeo mediante este protocolo. No se recomienda el uso de este sistema ya que es muy ineficiente. (Torres, 2009)

1.3.Estado del Arte.

En la actualidad el desarrollo de la tecnología *streaming* está en ascenso, con el avance de la televisión digital, sin embargo, no cumplen con las necesidades de los clientes debidos a deficiencias en el servicio, es por ello que el estudio de soluciones existentes relacionadas con este proceso permite sentar las bases en la construcción de nuevas soluciones adaptables a las solicitudes o demandas de los clientes. Para el diseño de servidores *streaming* distribuidos basados en clúster, específicamente en los elementos que componen la propuesta para el componente manejador de clúster la información existente es insuficiente, por lo que serán analizadas algunas variantes de servicio de video bajo demanda que permitan arrojar información útil para la construcción de dicho componente.

"Utilización de programación funcional distribuida y clústers Linux en el desarrollo de servidores de vídeo bajo demanda", realizado por M. Barreiro, V. M. Gulías, J. Mosquera, J. J. Sánchez pertenecientes al departamento de computación situado en la Universidad de Coruña, España. (M. Barreiro, 2001) El análisis de esta solución conduce a la conclusión que la propuesta necesita un refinamiento por:

Falta de Flexibilidad en el diseño: ya que la arquitectura jerarquita en tres niveles propuesta demostró en el proceso de prueba ser sofisticada para redes pequeñas, pero rígida para redes de topología más complicadas como una red de interconexion de redes metropolitanas.

Precencia de protocolos pesados: debido a la utilización de una distribucion *Progressive Download*[5] sobre el protocolo HTTP y TCP/IP. Debido a la capacidad de TCP a la hora de recibir información sin perdidas de integridad, una capacidad valiosa en otro tipo de transmisiones de datos, aunque en el caso del flujo de audio y/o video puede provocar repeticiones y retrasos, traduciéndose en una perdida de calidad a la hora de recibir el objeto multimedia por parte del usuario. Además, el protocolo HTTP no está diseñado para soportar largas transferencias de archivos.

Para el servidor ALLFRY'S se prevé para la trasmision del objeto multimedia desde el servidor hasta el cliente los protocolos RTP,RTSP y UDP, y para establecer la comunicación del componente manejador de clúster con otros componentes se utilizará el protocolo TCP debido a que el componente no trabaja con el objeto multimedia. El manejador está pensado para tener gran escalabilidad y disponibilidad, ya que está ideado para ofreser un servicio ininterrumpido. Además, permitirá amordarse con el menor coste posible a necesidades cambiantes.

"Un sistema de vídeo bajo demanda a gran escala tolerante a fallos de red", realizado por Javier A Balladini para optar por el grado de Doctor en la Universidad Autónoma de Barcelona, trabajo que propone la arquitectura de un sistema de video bajo demanda a gran escala (LVoD) distribuido en Internet, con comunicación unicast, denominada VoD-NTF (Video on Demand with Network Fault Recovery).

[5] ***Progressive download*** **(descarga progresiva o falso *streaming*):** método que se utilizada en un servidor que envía los archivos multimedia a través del protocolo HTTP y TCP/IP.

Tiene como fin garantizar ante fallos de red y caídas de servidores, la entrega del contenido multimedia a los clientes sin disminuir la calidad de los mismos y sin sufrir interrupciones durante su visualización. Permite ofrecer servicio de video bajo demanda verdadero (T-VoD), se explican los tres componentes que intervienen en una transmisión completa que involucran el alquiler y la visualización de un contenido multimedia, enfocando el análisis en el funcionamiento de los módulos que componen el servidor.

En la solución propuesta se explica cada módulo que compone la arquitectura interna del servidor, profundizando el estudio del módulo Control del Servidor de Video (ver figura 1). Este componente, de reducidas dimensiones, tiene la responsabilidad de coordinar la puesta en ejecución del servidor de video, y proporcionar un interfaz de comunicación centralizada para el proveedor del sistema. Desde él se puede configurar todos los aspectos del servidor de video, que abarcan a los demás componentes del servidor. Cuando llega una orden de iniciar el servidor, este componente pone en marcha cada uno de los módulos controlando que la ejecución de ellos sea correcta. (Balladini, 2008)

El módulo está conformado solamente por una clase, denominada video server, que crea exactamente un objeto de cada una de las clases asociadas. Cada uno de estos objetos creados, que ponen en marcha cada uno de los módulos del servidor, está previamente configurado con los parámetros que rigen el comportamiento particular de cada componente.

El módulo Control del Servidor de Vídeo de la propuesta y el Manejador de Clúster descrito por la arquitectura ADMS son similares en cuanto a funcionalidades, debido a que cada uno de diferentes formas pone en marcha los módulos que integran el servidor (ver figura 3). Además, el Manejador de Clúster debe ser capaz de determinar el ranking de medias dentro del servidor, proceso que muestra la demanda de los archivos de videos almacenados. La comunicación establecida entre el Manejador de Clúster y los componentes del servidor ALLFRY'S se forma a través del middlewore ICE mediante la comunicación cliente-servidor permitiendo explotar las cualidades distribuidas que esto admite.

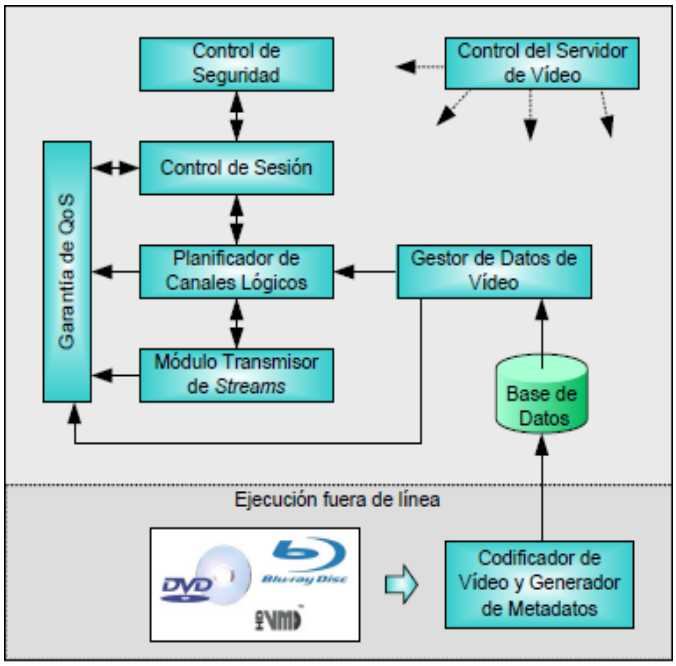


Figura 1. Arquitectura del servidor de video. (Balladini, 2008)

Después del análisis de la arquitectura VoD-NTF se concluye que es de vital importancia profundizar el estudio del proceso de migración o réplica dentro de un servidor *streaming* distribuido, enfatizando el proceso de migración del componente manejador de clúster dentro de un servidor *streaming* distribuido, así como la necesidad de realizar el proceso de calidad del servicio QoS.

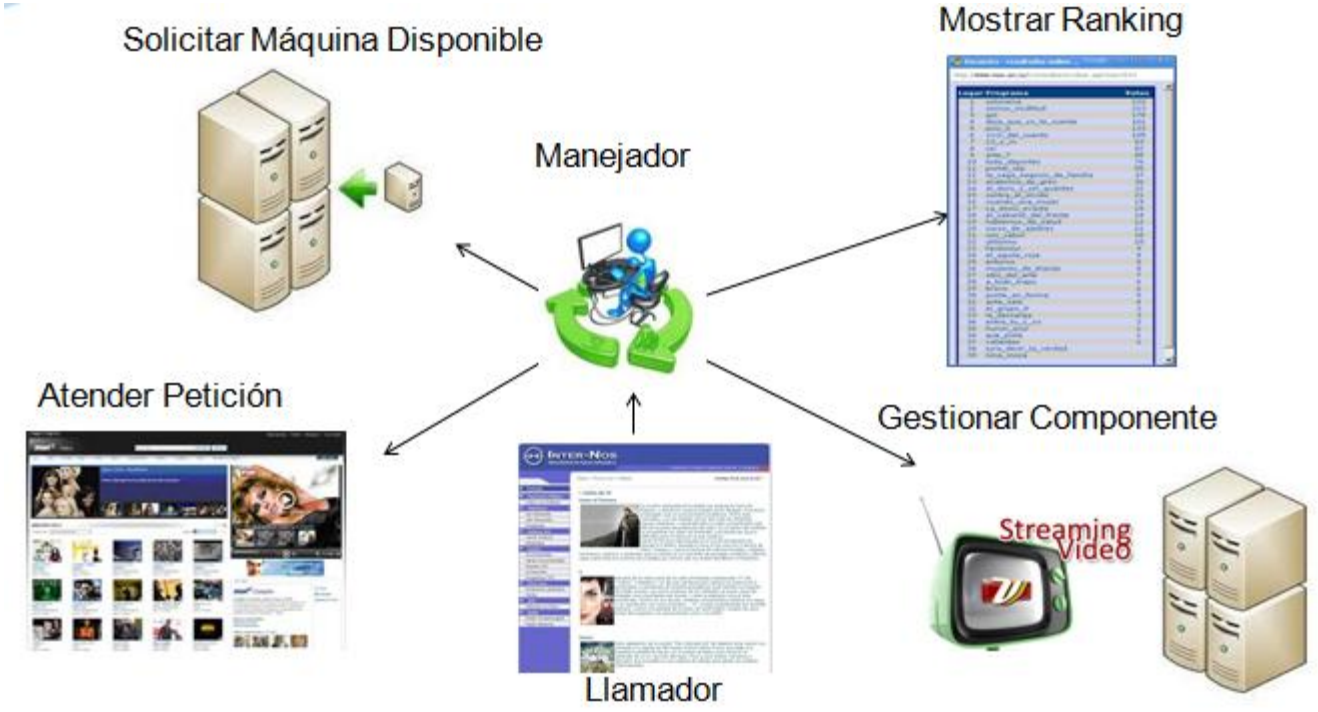


Figura 2. Funcionamiento del MC

Otras Soluciones

En los últimos años las empresas más importantes que comercializan productos multimedia han desarrollado sistemas relacionados con el *streaming*. Algunas de las soluciones son restrictivas, propietarias y comerciales como es el caso de *QuickTime Streaming Server de Apple; Helix Universal Server Basic 11.1.3 de RealNetworks* donde el número de flujos de vídeo soportados es directamente proporcional a la cantidad de dinero invertida en licencia; *Microsoft Windows Media Server* de *Microsoft* en donde dependiendo del tipo de licencia de *Windows* server adquirido, y de la cantidad de dinero invertida, se habilitan en mayor o menor grado las características o funcionalidades disponibles.

Otras de estas soluciones son de código abierto como es el caso de *Helix DNA Server RealNetworks* que comparte el mismo código base que la versión comercial mencionada anteriormente, abriendo solo el código de los componentes más básicos del servidor; el *Darwin Streaming Server* (DSS) que es la versión

de *QuickTime Streaming* Server entre otros. En general, el análisis en detalle de estos productos, conduce a la conclusión de que la mayoría representa soluciones caras, cerradas y no adaptables.

1.4.Conclusiones Parciales

Para la creación de este capítulo se manejaron conceptos asociados al dominio del problema que sirvieron de base para lograr un mejor entendimiento del tema de investigación. El análisis de las soluciones existentes para la transmisión de video sobre redes de computadora permitió a la autora evidenciar las características y tendencias actuales de los servidores *streaming*, demostrando la importancia de estos en la entrega de audiovisuales en Internet o en redes locales. El análisis de la arquitectura ADMS sirvió de apoyo para la búsqueda de la solución al problema científico planteado en este trabajo de diplomado.

Capítulo 2: Tendencia y tecnologías actuales.

2.1. Introducción.

En el presente capítulo se exponen las tecnologías, herramientas y metodologías utilizadas en el mundo para la construcción de aplicaciones de escritorio que se ajustan al desarrollo de la solución que se propone. Se tuvieron en cuenta tres aspectos fundamentales a la hora de realizar la selección: que dichas metodologías, herramientas y tecnologías fueran las que mejor se ajustaran al diseño arquitectónico del sistema, que cumpliera las expectativas del cliente y que pudieran ser desplegadas sobre la plataforma GNU/Linux.

2.2.Metodologías de desarrollo de *software*.

Proceso Unificado de Desarrollo de *Software* (RUP)

Posee como directriz producir *software* que cumplan con la calidad requerida por el usuario, orientado por una buena planificación dentro de un presupuesto establecido. Consta de cuatro fases de confección: inicio, elaboración, construcción y transmisión, las cuales se desarrollan mediante iteraciones, en cada una se reproduce el ciclo de vida en cascada a menor escala, teniendo como objetivo la evaluación de las iteraciones antecedentes. Además de las fases, el ciclo de vida de esta metodología contiene flujos de trabajos, los cuales se dividen en flujos de trabajo de desarrollo y flujos de trabajo de soporte. (Pressman, 2002)

RUP es una metodología basada en componente, por lo que el *software* va a estar conformado por uno o varios componentes que interactúan entre sí a través de sus interfaces. Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del *software*. Utiliza UML[6] como lenguaje de notación y entre sus principales elementos se encuentra los trabajadores (quién), las actividades (cómo), los artefactos (qué) y el flujo de trabajo (cuando). Es una metodología que provee una planificación eficiente, organizada y detallada del proceso de desarrollo, proporcionando un *software* que cumple con los estándares internacionales de calidad, atendiendo a las necesidades del cliente. Está dirigido por:

[6] **Lenguaje Unificado de Modelado(UML)**

- Casos de uso
- Centrado en la arquitectura
- Iterativo e incremental

La metodología definida para dirigir el proceso de desarrollo del componente Manejador de Clúster es RUP ya que es dirigido por casos de uso, centrado en la arquitectura e iterativo e incremental. Además permite desarrollar aplicaciones sacando el máximo provecho de las nuevas tecnologías, mejorando la calidad, el rendimiento, la reutilización, la seguridad y el mantenimiento del *software* mediante una gestión sistemática de los riesgos. Aprueba la producción de *software* que cumpla con las necesidades de los usuarios, a través de la especificación de los requisitos. Admite llevar a cabo el proceso de desarrollo práctico, brindando amplias guías, plantillas y ejemplos para todas las actividades críticas.

2.3.Lenguaje de Modelado.

Es un conjunto estandarizado de símbolos y de modos que se utiliza para modelar parte de un diseño de *software* orientado a objetos. Algunas organizaciones los usan extensivamente en combinación con una metodología de desarrollo de *software* para avanzar de una especificación inicial a un plan de implementación y para comunicar dicho plan a todo un equipo de desarrolladores. El uso de un lenguaje de modelado es más sencillo que la auténtica programación.

Lenguaje Unificado de Modelado (UML)

El Lenguaje de Modelado Unificado es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de *software*. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero está pensado

para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos. **(Jacobson, y otros, 2000)**

Para el desarrollo del componente Manejador de Clúster se prevé la utilización de UML como lenguaje de modelado, debido a que permite modelar sistemas utilizando técnicas orientadas a objetos. Permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y complejos. No es difícil de utilizar ni de aprender. Permite crear diagramas que representan alguna parte o puntos de vista del sistema.

2.4.Herramientas CASE.

Se puede definir a las Herramientas CASE[7] como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de *software* y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un *Software*. **(Scribd, 2008)**

Visual Paradigm para UML.

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El *software* de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. (Alonso E. M., 2009)

Cuenta con una licencia gratuita y comercial, soporta aplicaciones web, varios idiomas, genera código para Java y exportación como HTML, fácil de instalar y actualizar, gran compatibilidad entre ediciones. Se ha actualizado rápidamente en sintonía con el nuevo desarrollo de técnicas de modelado UML 2.1 con el propósito de generar un entorno de modelados visuales en el que se reúnen hoy todas las necesidades tanto de *software* y tecnología, como de las necesidades de comunicación.

Fue definido Visual Paradigm como herramienta CASE porque utiliza UML 2.1 como lenguaje de modelado, es compatible con la metodología seleccionada y posibilita la generación de código y

[7] **CASE: (Computer-Aided Systems Engineering)**

documentación. Es multiplataforma. Ofrece un diseño centrado en casos de uso y enfocado al negocio que generan un *software* de mayor calidad. Permite dibujar 13 tipos de diagramas diferentes a través de un intuitivo modelado visual y la aplicación de ingeniería inversa. Además, posee una amplia bibliografía tanto en materiales audiovisuales como documentación digital. Es muy sencillo de usar, fácil de instalar y actualizar.

2.5.Lenguajes de Programación

Las máquinas en general, y las computadoras en particular, necesitan de un lenguaje propio para poder interpretar las instrucciones que se les dan y controlar su comportamiento. Ese lenguaje que permite esta relación con las computadoras es el lenguaje de programación. En la actualidad existe una amplia variedad de lenguajes de alto nivel para el desarrollo de aplicaciones.

C++.

Es un lenguaje de programación de alto nivel orientado a objetos. Permite la programación estructurada. Mantiene las ventajas del C en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Es un lenguaje de gran velocidad en la ejecución de código. Se puede utilizar tanto para escribir *software* de bajo nivel, como *drivers* y componentes de sistemas operativos, como para el desarrollo rápido de aplicaciones, según el marco de trabajo con el que se disponga. Los compiladores de C++ generan código nativo con un alto grado de optimización en memoria y velocidad, lo que lo convierte en uno de los lenguajes más eficientes. (Rivera, 2008)

Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores). Es muy potente en lo que se refiere a creación de sistemas complejos y es un lenguaje sólido. Existen muchos entornos de programación para C++. No existen estándares para ello. El sistema a desarrollar debe ser lo más ligero y rápido posible dado a la cantidad de peticiones que debe procesar en el menor tiempo de ejecución. Para lograr estos se prevé la utilización de C++ debido a que es un lenguaje de rápida ejecución, potente y versátil.

2.6.Entorno de Desarrollo Integrado (IDE)

Un Entorno de Desarrollo Integrado o IDE (del inglés *Integrated Development Environment*) es un programa compuesto por un conjunto de herramientas que proveen un marco de trabajo amigable para los lenguajes de programación. Dentro de las partes fundamentales que lo integran se encuentra el editor de código, compilador, depurador y un constructor de interfaz gráfica de usuario.

Para poder utilizar el lenguaje de programación C++ es necesario el uso de un Entorno de Desarrollo Integrado. Por lo que se prevé la utilización de Qt Creator debido a que es un IDE bastante completo, moderno, potente, fácil de manejar, eficiente, abierto y gratuito, que permite el desarrollo rápido de aplicaciones. Es compatible con los sistemas operativos Windows, Linux y Mac OS X. Utiliza el lenguaje C++ de forma nativa por lo que proporciona completamiento de código y ayuda de contexto explotando al máximo las ventajas que ofrece la utilización de este lenguaje de programación. Con su utilización se obtiene un desarrollo más rápido de las aplicaciones. Permite realizar programación visual y dirigida por eventos.

Contiene soporte para la construcción de aplicaciones de escritorio como diseñador de formularios, herramienta para proyectos y administración, depurador visual, resaltado y autocompletado de código. Se basa en Qt, una librería multiplataforma y gratuita para la creación de interfaces gráficas, programación web, multi-hilos y bases de datos. Con su utilización se obtiene un desarrollo más rápido de las aplicaciones.

2.7.*Middleware*.

Se puede entender un *middleware* como un *software* de conectividad que hace posible que aplicaciones distribuidas pueden ejecutarse sobre distintas plataformas heterogéneas, es decir, sobre plataformas con distintos sistemas operativos, que usan distintos protocolos de red y, que incluso, involucran distintos lenguajes de programación en la aplicación distribuida. Desde otro punto de vista distinto, un *middleware* se puede entender como una abstracción en la complejidad y en la heterogeneidad que las redes de comunicaciones imponen. De hecho, uno de los objetivos de un *middleware* es ofrecer un acuerdo en las interfaces y en los mecanismos de interoperabilidad, como contrapartida de los distintos desacuerdos en *hardware*, sistemas operativos, protocolos de red, y lenguajes de programación. (Fernández, 2006)

El *middleware* es una capa de comunicación que debe proporcionar de forma transparente a los programadores métodos eficaces que permita la construcción de aplicaciones distribuidas. Es el *software* que actúa entre el sistema operativo y las aplicaciones y que brinda al usuario la experiencia de estar utilizando una única máquina. Optimiza el sistema y provee herramientas de mantenimiento para procesos pesados como podrían ser migraciones, balanceo de carga, tolerancia de fallos, entre otros. (Solingest, 2007)

La heterogeneidad en los sistemas distribuidos es resuelta por medio de componentes denominados *middleware*s, se ejecuta sobre diversas combinaciones de arquitectura de procesador y sistema operativo y supera esta variedad ofreciendo igual funcionalidad en todas las computadoras por medio de una biblioteca de programas o API, útil para el desarrollo de aplicaciones distribuidas. (Escobar, 2007)

ICE [8]

Es un *middleware* que proporciona herramientas, APIs, y soporte de bibliotecas para construir aplicaciones cliente–servidor orientado a objeto. Las aplicaciones de ICE son adecuadas para uso en entornos heterogéneos: donde cliente y servidor pueden ser escritos en lenguajes de programación diferentes, puedan correr sobre sistemas operativos diferentes, y puedan comunicarse usando una variedad de tecnologías de red. (Fernández, 2006)

Terminología Cliente-Servidor

Los términos cliente y servidor no están directamente asociados a dos partes distintas de una aplicación, sino que más bien hacen referencia a los roles que las diferentes partes de una aplicación pueden asumir durante una petición: (Fernández, 2006)

- Los clientes son entidades activas, es decir, emiten solicitudes de servicio a un servidor.
- Los servidores son entidades pasivas, es decir, proporcionan un servicio en respuesta a las solicitudes de los clientes.

Normalmente, los clientes no son clientes puros en el sentido de que sólo solicitan peticiones. En su lugar, los clientes suelen ser entidades híbridas que asumen tanto el rol de cliente como el de servidor.

[8] ***Internet Communication Engine*(ICE)**

Para establecer la comunicación entre el Manejador de clúster y otras aplicaciones, se hará uso del *middleware* ICE, debido a que admite su desarrollo en Linux, permite el manejo de sistemas distribuidos de forma simple, apoya el envío asincrónico de mensajes invocación (AMI[9]) y el tratamiento asíncrono de mensajes (AMD[10]). Además, contempla la réplica y el balanceo de carga.

2.8.Conclusiones Parciales

Luego de haber analizado metodologías, tecnologías, lenguajes de programación y herramientas utilizadas en el mundo para el desarrollo de *software* se realizó una selección de estas para construir la solución propuesta, para ello se tuvo en cuenta como principales elementos a considerar que fueran libres o de código abierto y que pudieran ser desplegada sobre la plataforma GNU/Linux, ya que en Cuba actualmente se aboga por la soberanía e independencia tecnológica y la UCI no se encuentra exenta de esto. El correcto empleo de este conjunto de técnicas contribuye a obtener un *software* con calidad. Después de haber hecho una completa selección de las tecnologías y herramientas para desarrollar el componente, se está en condiciones de realizar una presentación de la solución propuesta.

[9] Invocación de métodos asíncrona o AMI (Asynchronous Method Invocation)
[10] Tratamiento de métodos asíncrono o AMD (Asynchronous Method Dispatch)

Capitulo 3: Presentación de la solución propuesta.

3.1.Introducción

En este capítulo se confecciona el modelo del negocio en el que aparece enmarcado la propuesta del componente a desarrollar. Se definen los requerimientos funcionales y no funcionales del componente, se especifican los actores y casos de uso del sistema. Además se expone la definición del modelo de análisis y el diseño del sistema. En dicho capítulo se determinan las clases persistentes en la aplicación.

3.2.Propuesta de sistema.

Después del estudio realizado y del exhaustivo análisis del objeto de estudio, se ha decidido proceder con el desarrollo de un componente capaz de procesar un conjunto de peticiones para el almacenamiento y la trasmisión de medias así como la interacción entre módulos, facilitando en su totalidad el procesamiento de las peticiones de servicio dentro del servidor *streaming* distribuido. Se prevé la implementación de un Manejador de Clúster que dado una solicitud de servicio sea capaz de atenderla de forma correcta, para ello debe clasificar la petición según el tipo (almacenamiento o *streaming*). Además debe ser capaz de distinguir qué módulo dará solución a la petición y habilitarlo en un nodo disponible del clúster. También el manejador de clúster debe determinar cuál es la media más solicitada por los usuarios. Asimismo estable la réplica de los componentes Colector y Distribuidor de Datos cuando exista algún incombeniente que afecte el nodo donde se encucentran atendiendo alguna petición uno de estos componentes.

3.3.Modelo del Negocio.

Unos de los flujos más significativos durante la fase de inicio en el desarrollo del *software* es el modelado del negocio el cual tiene como principales objetivos:

- Comprender la estructura y la dinámica de la organización en la cual se va a implantar un sistema
- Comprender los problemas actuales de la organización e identificar las mejoras potenciales.
- Asegurar que los clientes, los usuarios finales y desarrolladores tengan un mejor entendimiento común de la organización.

- Derivar los requerimientos del sistema que va a soportar la organización.

Dependiendo de la situación o escenario, existen diversas alternativas para el desarrollo de este proceso, no siempre es necesaria o posible la total realización de un modelo completo del negocio. Si se determina que no es necesario un modelado completo del negocio, se realizará lo que se conoce como modelo de dominio. (Booch, 2000)

3.4.Modelo del Dominio.

Este modelo ayuda a comprender los términos que utilizan los usuarios, los conceptos con los que trabajan los desarrolladores y con los que deberá trabajar el componente. El objetivo del modelo de dominio es comprender y describir las clases más importantes dentro del contexto del sistema, en otras palabras el modelado del dominio deberá contribuir a una comprensión del problema que el sistema resuelve en relación a su contenido. (Booch, 2000) En la figura 3 se muestra el diagrama del modelo del domino del componente Manejador de Clúster. En este modelo presenta un sistema que realiza una petición de servicio al componente Manejador de clúster. El manejador es el encargado de identificar la petición si es de almacenamiento o streaming, y en dependencia del tipo de petición habilita un Colector o Distribuidor de Datos que atenderá esa petición en un nodo del clúster que será obtenido del componente Adaptador de Núcleo. Además, muestra una lista con las medias de mayor demanda almacenadas en el servidor.

Glosario de términos para identificar todos los conceptos que se utilizarán en el diagrama:

Petición: Solicitud de un servicio determinado por parte del cliente.

Sistema: Es el usuario o sistema que se beneficia de las utilidades del componente.

Ranking: Es la demanda de las medias almacenadas en el servidor.

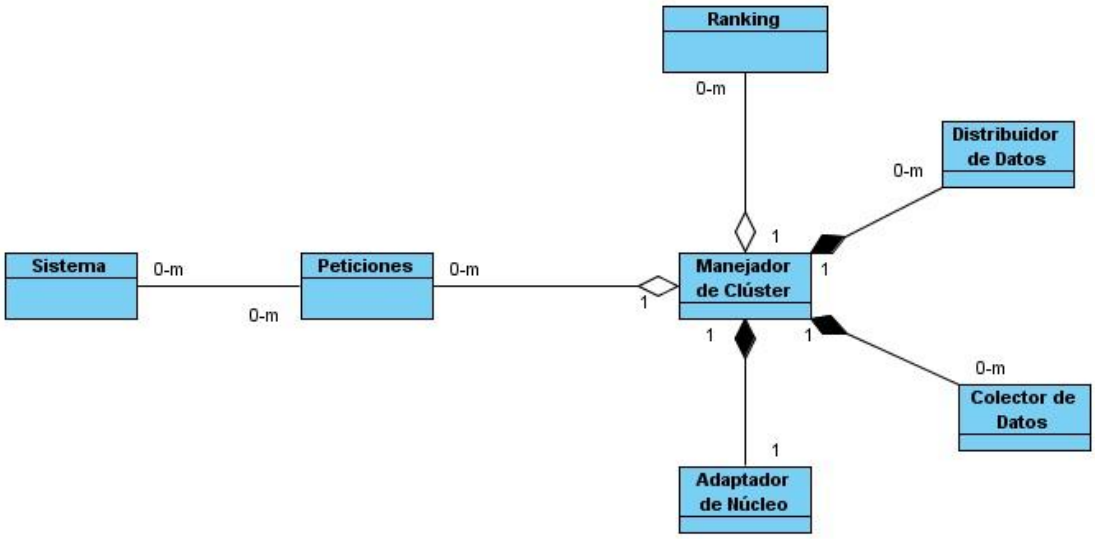


Figura 3. Modelo del Dominio

3.5.Especificación de los requerimientos del *software*.

Un requerimiento es una funcionalidad que puede ser alcanzada o poseída por un sistema o un componente de un sistema (es decir, las condiciones o capacidades que el sistema debe cumplir) (Booch, 2000), para satisfacer un contrato u otro documento impuesto formalmente entre los desarrolladores del *software* y los clientes del mismo. Define qué es lo que el sistema debe hacer. Especifican propiedades del sistema, como restricciones del entorno o de la implementación, rendimiento, dependencia de la plataforma entre otros. (Booch, 2000)

3.5.1. Requisitos Funcionales.

RF.1-Atender peticiones.

1.1-Identificar el tipo de petición: el sistema debe identificar si la petición es de almacenamiento o de *streaming.*

1.2-Solicitar direcciones ip para satisfacer la petición: el sistema debe solicitar al adaptador de núcleo un nodo que cumpla las condiciones para satisfacer la petición.

1.3-Identificar el componente que debe satisfacer la petición: el sistema debe identificar entre el Colector o el Distribuidor que componente debe atender la petición.

1.4-Activar un componente: el sistema de activar un componente Colector o Distribuidor en un nodo del clúster.

RF2- Determinar entre las medias almacenadas las más populares.

3.5.2. Requisitos no Funcionales

- *Software*: el sistema requiere para su uso de un nodo que tenga instalado como SO *Ubuntu* 10.10 y la librería ICE.
- *Hardware:* Se requiere tarjeta de red. Memoria RAM de 512MB o superior. Disco duro de cualquier capacidad. Procesador 3 GHz o superior.
- Usabilidad: el componente podrá ser usado por un sistema que instancie al manejador de clúster para atender una petición o utilizar cualquiera de sus funcionalidades.
- Confiabilidad: la herramienta de implementación a utilizar debe tener soporte para recuperación ante fallos y errores.

3.5.3. Diagrama de Casos de uso del Sistema.

Después de obtenido los requerimientos funcionales con los que el componente Manejador de clúster debe contar para satisfacer las necesidades del cliente, fue desarrollado el diagrama de casos de uso del sistema mostrado en la figura 4. En el diagrama se muestra la relación entre los actores y los casos de uso del sistema.

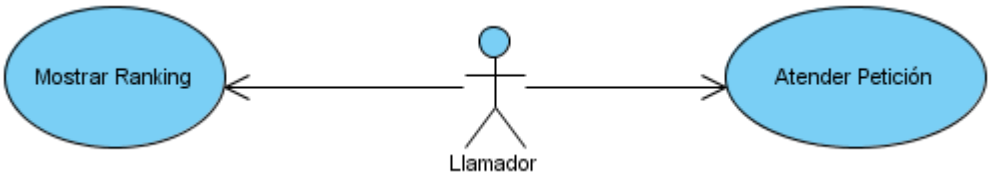


Figura 4. Diagrama de CU Sistema

3.5.4. Definición de Actores del Sistema.

Cada trabajador del negocio (inclusive si fuera un sistema ya existente) que tiene actividades a automatizar es un candidato a actor del sistema. Si algún actor del negocio va a interactuar con el

sistema, entonces también será un actor del sistema. Los actores del sistema: no son parte de él, pueden intercambiar información con él, pueden ser un recipiente pasivo de información y pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado. (Booch, 2000) En la tabla 1 se define el actor del sistema Manejador de Clúster.

Tabla 1. Definición del Actor del Sistema

Actores	**Justificación**
Llamador	Es la aplicación o sistema que inicializa todos los casos de uso del componente manejador de clúster.

3.5.5. Expansión de los Casos de Uso del Sistema.

Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema. (Booch, 2000) Por lo planeado anteriormente se puede decir que un caso de uso es una secuencia de acciones que el sistema lleva a cabo para ofrecer algún resultado de valor para un actor.

Tabla 2.Atender Petición.

Caso de uso	
CU-1	Atender Petición
Propósito	Atender de forma correcta una solicitud realizada al sistema.
Actores: Llamador	
Resumen: Caso de uso que se encarga de dada una petición de servicio catalogarla (Almacenamiento o *Streaming*).Además, debe habilitar un Colector de Datos o un Distribuidor de Datos para satisfacer la petición en un nodo disponible del clúster.	
Referencias	RF 1
Precondiciones	El usuario tiene que realizar una petición.

Acción del actor	**Respuesta del sistema**
1. El caso de uso comienza cuando el llamador envía una solicitud de servicio al sistema.	1.1. El sistema debe identificar el tipo de petición. 1.2. Solicita al Adaptador de Núcleo la dirección de un nodo disponible para atender la petición. 1.3. Si la petición es de Almacenamiento el sistema debe activar un Colector de Datos (ver sección1), si la petición es de *Streaming* el sistema debe activar un Distribuidor de Datos (ver sección2).
Sección1: Activar Colector de Datos	
Acción del Actor	**Respuesta del sistema**
	1.4. Si la petición es de Almacenamiento el sistema debe Activar en la dirección ip enviada por el Adaptador de Núcleo el componente Colector de Datos.
Sección2: Activar Distribuidor de Datos	
Acción del Actor	**Respuesta del sistema**
	1.5. Si la petición es de Streaming el sistema debe Activar en la dirección ip enviada por el Adaptador de Núcleo el componente Distribuidor de Datos. 1.6. El sistema actualiza la lista de las medias más populares.

Tabla 3. Mostrar Ranking.

Caso de uso		
CU-2	Mostrar Ranking.	
Propósito	Determinar entre las medias almacenadas cuáles son las más solicitadas	
Actores: Llamador		
Resumen: El caso de uso se inicializa cuando el actor solicita la lista de las medias con más demandas almacenadas en el servidor. El sistema muestra una lista ordenada de mayor a menor teniendo en cuenta la cantidad de veces que ha sido solicitada esa media para ser transmitida.		
Referencias	RF 2	
Acción del actor		**Respuesta del sistema**
1. El caso de uso comienza cuando el actor realiza una solicitud para visualizar la lista de medias más populares.		1.1. Muestra de mayor a menor una lista con el nombre de la media y la cantidad de veces que ha sido solicitada por un usuario para ser visualizada.

3.6.Diagramas de Clase del Diseño

Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Se utilizan para modelar la vista de diseño estática de un sistema. Además son importantes no sólo para visualizar, especificar y documentar modelos estructurales, sino para construir sistemas ejecutables, aplicando ingeniería directa e inversa.

Los Diagramas de clases del diseño (DCD) representan las especificaciones de las clases e interfaces software en una aplicación. (Larman, 1999) Los DCD son una representación más concreta y detallada que los diagramas de clases del análisis, aunque también representan la parte estática del sistema conteniendo las clases y sus relaciones. A continuación se muestra en la figura 5 el diagrama de clases para el caso de uso Atender Petición, de forma tal que se facilite la compresión de las relaciones entre los

distintos componentes. Para ver los demás diagramas relacionado con otros casos de uso dirigirse al ANEXO1.

3.6.1. Patrones de Diseño

El patrón es una solución probada que se puede aplicar con éxito a un determinado tipo de problema que aparece repetidamente (Canal, 2005). Por lo que se puede decir que el patrón es una solución a un problema en un contexto, permiten reutilizar soluciones a problemas comunes y son un esqueleto básico que cada persona adapta a las peculiaridades de su problema.

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de *software* y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. El patrón de diseño identifica las clases e instancias participantes en el diseño del *software*, sus roles y colaboraciones, y la distribución de responsabilidades. Cada patrón de diseño se centra en un problema concreto, describiendo cuándo aplicarlo y si tiene sentido hacerlo teniendo en cuenta otras restricciones de diseño, así como las consecuencias, ventajas e inconvenientes de su uso.

En el diseño del componente Manejador de Clúster para asignar las responsabilidades se tuvieron en cuenta algunos de los Patrones GRASP[11] como el Experto, Alta cohesión y Bajo Acoplamiento. La responsabilidad es la obligación que tienes una clase de hacer o conocer algo. (Larman, 1999)

Experto: Consiste en asignar una responsabilidad al experto en información, en otras palabras es la clase que tiene la información necesaria para realizar la responsabilidad. Se utiliza en clases como Ranking, CE Petición y CE pAlmacenamiento que se encargar de dar a conocer los atributos privados de cada clase.

Alta cohesión: Una clase tiene una responsabilidad moderada en un área funcional y colabora con otras clases para llevar a cabo las tareas. Ejemplo: La clase Distribuidor de Datos que se relaciona con la clase Consumidora Distribuidor para realizar la acción de activar un componente de forma remota.

[11] **GRASP: patrones generales de software para asignar responsabilidades.**

Bajo Acoplamiento: Asigna una responsabilidad de manera que el acoplamiento permanezca bajo. Ejemplo: La clase pAlmacenamiento hereda solo de la clase Petición para lograr un bajo acoplamiento entre estas clases.

Además, de los patrones antes mencionados se tuvo en cuenta algunos de los patrones GoF[12] entre los que se pueden encontrar:

Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Fue utilizado con el objetivo de realizar una única instancia de la clase CC_Manejador.

Fachada: Tiene como objetivo proporcionar una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema. El Manejador de Clúster utiliza una clase interfaz Publicador ICE que abarca todas las funcionalidades del componente y a través de esta se realiza toda la interacción con los otros componentes o aplicaciones

Observador: Define una dependencia "uno-a-muchos" entre objetos, para que, cuando uno de ellos cambie su estado, todos los que dependan de él sean avisados y puedan actualizarse convenientemente. Dado que el "observado" no conoce la clase de "observadores" que tiene, el acoplamiento que existe es mínimo y abstracto. Por ello, tanto unos como otros pueden variar y evolucionar en diferentes niveles de detalle de forma independiente. En el Manejador se utiliza pues hay clases de la capa del negocio que dependen de la clase de ICE en la capa superior (Interfaz) y es necesario que esta notifique en caso de algún cambio.

3.6.2. Patrones de Arquitectura

Los patrones de arquitectura se centran en la arquitectura del sistema. Definen una estructura fundamental sobre la organización del sistema. Proveen un conjunto predefinido de subsistemas, cuáles son sus responsabilidades y cómo se interrelacionan. Estos patrones podrían considerarse estrategias de alto nivel que abarcan componentes a gran escala, propiedades y mecanismos del sistema. En el diseño del componente además de los patrones antes mencionados se utilizó el Patrón de Arquitectura en Capas.

El patrón Capas: se relaciona con la arquitectura lógica; es decir, describe la organización conceptual de los elementos del diseño en grupos, independiente de su empaquetamiento o despliegue físico. (Larman,

[12] GoF (*Gang of Four o* "pandilla de los cuantos") conjunto de patrones descritos en el libro Design Patterns.

1999) Este tipo de patrón cada capa invoca solo al nivel inferior. En el Manejador de Clúster se estructuro el trabajo en tres capas ver las figura 6 donde se puede encontrar la Capa de Interfaz, Negocio y Acceso a Datos.

Diagrama de CD Atender Petición

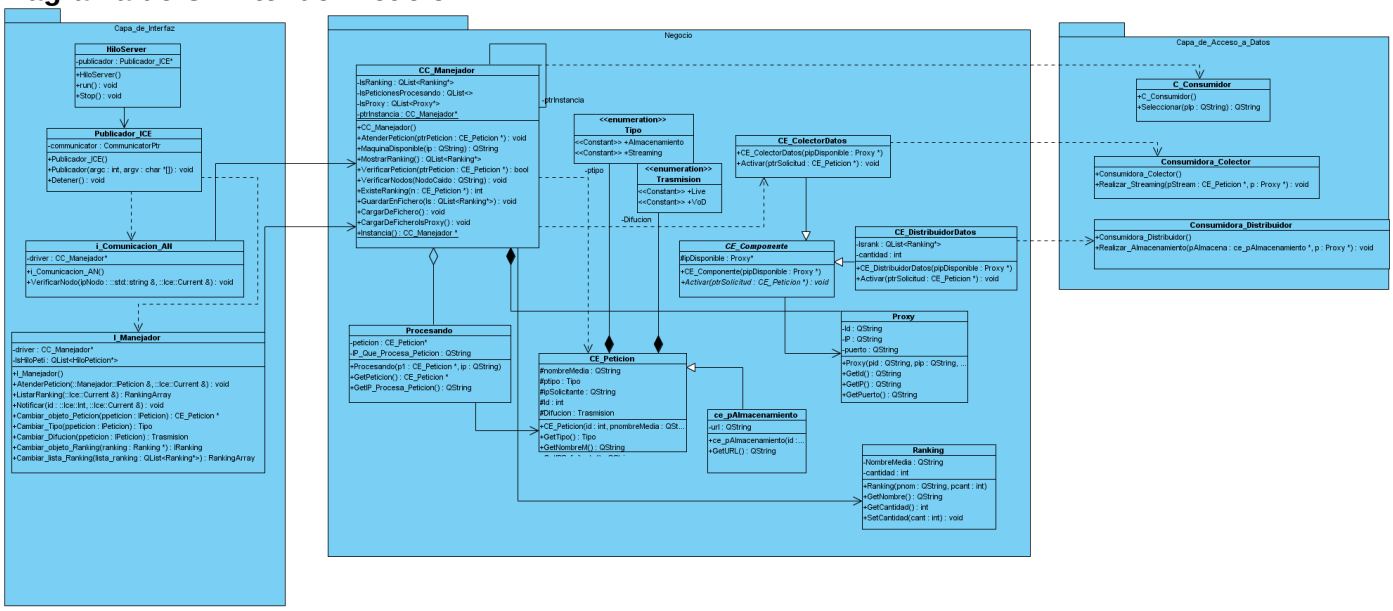


Figura 5. Diagrama de CD Atender Petición.

3.7.Conclusiones Parciales

Como resultado de este capítulo se generaron los artefactos más importantes de los primeros flujos de trabajo propuestos por RUP. Debido a que no se cuenta con un proceso de negocio bien definido, se realizó el Modelo de Dominio, el cual posibilitó un mejor entendimiento del entorno del sistema. Se realizó un levantamiento de requisitos funcionales y no funcionales, que ayudó a precisar qué es lo que el sistema debe hacer para satisfacer las necesidades del cliente. Se modeló el Diagrama de Casos de Uso del Sistema, describiendo al Actor y Casos de Uso que forman parte de este, todo esto permite contar con una documentación organizada de las restricciones requeridas para el desarrollo del componente Manejador de Clúster. Después de desarrollados los diagramas de clases del diseño perteneciente a cada caso de uso se procederá a la implementación de las clases definidas en el mismo, teniendo en cuenta las restricciones del diseño que tiene impacto en el sistema. De esta forma queda presentada la solución propuesta y se crean las condiciones para pasar a la fase de construcción del componente.

Capítulo 4: Implementación y prueba de la solución propuesta.

4.1.Introducción.

En este capítulo se lleva a cabo la construcción del sistema, partiendo de los resultados obtenidos en la fase de diseño, así como las pruebas del mismo. Se realiza el Modelo de Implementación que está compuesto por los diagramas de componentes y los diagramas de despliegue. También recoge el modelo de prueba.

4.2.Modelo de Implementación

El modelo de implementación describe cómo los elementos del diseño se implementan en términos de componentes, ficheros de código fuente, ejecutables entre otros. (Booch, 2000) En el modelo de implementación se empieza con los resultados obtenidos en el diseño y se crean los componentes físicos de la aplicación que se traducen en ficheros de código fuente .cpp y .h debido a su implementación en el lenguaje C++. Dentro del modelo de implementación se encuentran el Diagrama de Componentes y el Diagrama de Despliegue.

4.2.1. Diagrama de Componentes

Los Diagramas de Componentes modelan la vista estática de un sistema. Se representa como un grafo de componentes software unidos por medio de relaciones de dependencia (compilación, ejecución), pudiendo mostrarse las interfaces que estos soporten. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes, por lo cual cada diagrama describe un apartado del sistema. En la figura 6 se muestra el Diagrama de Componentes del código fuentes para el Manejador de Clúster donde se evidencia los ficheros .h y .cpp de cada componente debido al desarrollo en C++.

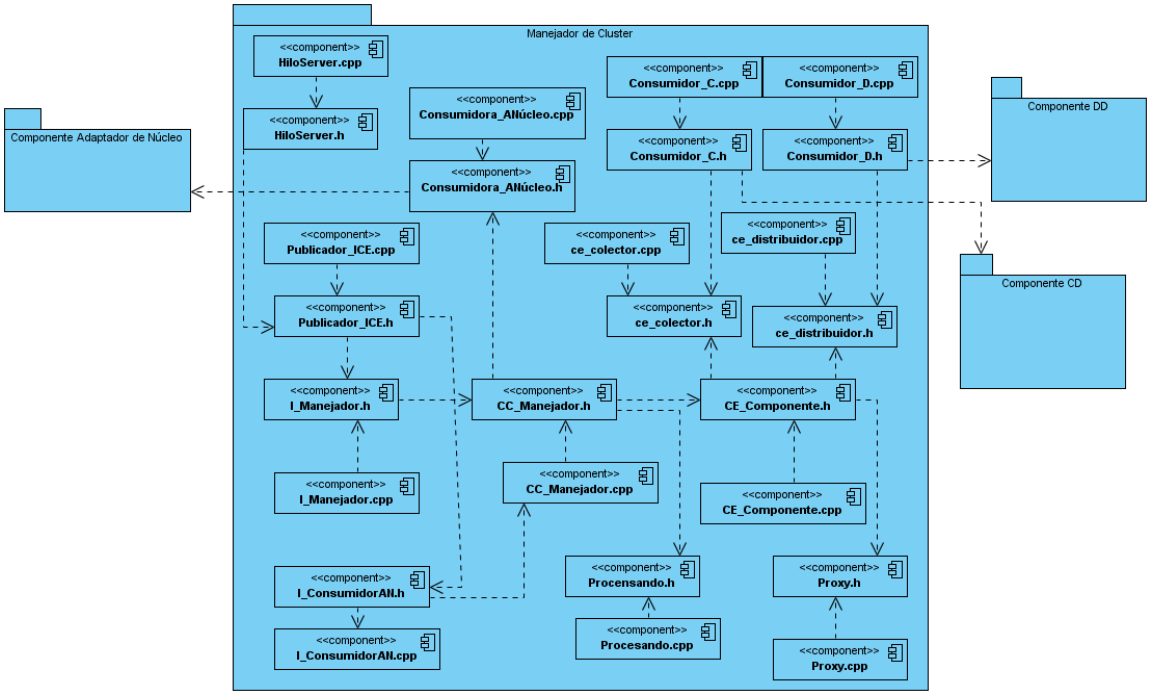


Figura 6. Diagrama de Componente MC.

4.3.Prueba

Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente. La prueba de *software* es un elemento crítico para la garantía de la calidad del *software* y representa una revisión final de las especificaciones del diseño y de la codificación. Este proceso permite la detección de los errores y asegura que las entradas definidas producen resultados reales de acuerdo con los requeridos por el cliente.

Cualquier producto de ingeniería puede probarse de una de las dos formas:

Pruebas de caja negra:

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del *software*, por lo que los casos de prueba pretenden demostrar que las funciones del *software* son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener

mucho en cuenta la estructura interna del *software*. Se centran principalmente en los requisitos funcionales del *software*. Las pruebas de caja negra permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. (Pressman, 2002)

Pruebas de la caja blanca:

Requieren del conocimiento de la estructura interna del programa y son derivadas a partir de las especificaciones internas de diseño o el código. Se basan en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del *software* proponiendo casos de pruebas que ejerciten conjunto específicos de condiciones y/o bucles, se pueden examinar varios fragmentos de códigos en diferentes puntos del programa para comprobar si el estado obtenido coincide con el esperado o mencionado. (Pressman, 2002)

Mediante los métodos de prueba de la caja blanca, el ingeniero de *software* puede obtener casos de prueba que garanticen que:

- Se ejerciten por lo menos una vez todos los caminos independientes para cada módulo.
- Se ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsa.
- Ejecuten todos los bucles en sus límites y con sus límites operacionales.
- Se ejerciten las estructuras internas de datos para asegurar su validez.

Debido a que la aplicación no cuenta con una interfaz visual con la que el usuario pueda interactuar no se le pueden hacer pruebas de caja negra al componente, por lo que serán aplicadas la prueba de caja blanca mediante el método del camino básico.

4.3.1. Prueba de caja blanca

La prueba del camino básico:

La prueba del camino básico es una técnica de prueba de caja blanca. Permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática. Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.

2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

En la figura 8 se muestra una fracción del código correspondiente al caso de uso Atender Petición al cual le serán aplicadas las pruebas de caja blanca mediante la técnica del camino básico antes mencionada. Para llevar a cabo esta técnica fue necesario realizar la conversión de construcción estructurada (fragmento de código) al grafo de flujo. (Ver la figura 9) Estas modificaciones se produjeron enumerando cada sentencia del código y convirtiéndolas a la notación del Grafo de Flujo. (Ver figura 7)

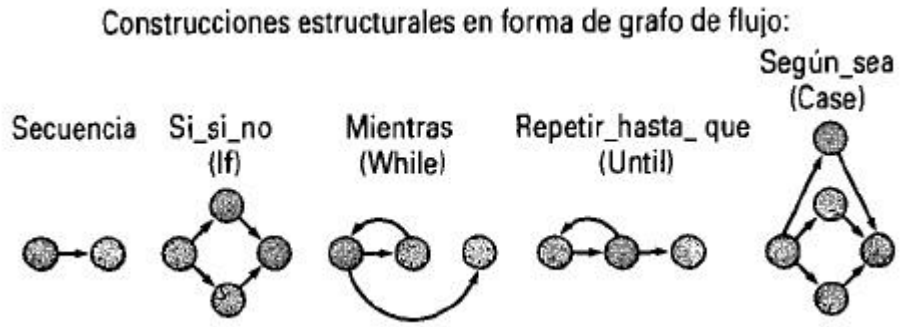


Figura 7 Notación de grafos de flujo para las instrucciones (Pressman, 2002)

Figura 8 Funcionalidad Atender Petición

Un Grafo de Flujo está formado por 3 componentes fundamentales que ayudan a su elaboración, comprensión y brinda información para confirmar que el trabajo se está haciendo adecuadamente.

Los componentes son:

- Nodo.

Cada círculo representado se denomina nodo del Grafo de Flujo, el cual representa una o más secuencias procedimentales. Un solo nodo puede corresponder a una secuencia de procesos o a una sentencia de decisión. Puede ser también que hayan nodos que no se asocien, se utilizan principalmente al inicio y final del grafo.

- Aristas.

Las flechas del grafo se denominan aristas y representan el flujo de control, son análogas a las representadas en un diagrama de flujo. Una arista debe terminar en un nodo, incluso aunque el nodo no represente ninguna sentencia procedimental.

- Regiones.

Las regiones son las áreas delimitadas por las aristas y nodos. También se incluye el área exterior del grafo, contando como una región más. La cantidad de regiones es equivalente a la cantidad de caminos independientes del conjunto básico de un programa.

Después de obtenido el Grafo de flujo para el fragmento de código de la figura 9 antes ilustrado se está en condiciones de calcular para este grafo la complejidad ciclomática. La complejidad ciclomática es una métrica del *software* que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez. *(Pressman, 2002)*

Se entiende por caminos independientes cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias de proceso o una nueva condición. En términos del grafo de flujo, un camino independiente está constituido por lo menos por una arista que no haya sido recorrida anteriormente a la definición del camino. *(Pressman, 2002)*

Existen 3 formas de calcular la complejidad ciclomática:

1. El número de regiones del grafo de flujo coincide con la complejidad ciclomática.

2. La complejidad ciclomática, V(G), de un grafo de flujo G se define como:
   V (G) = A - N + 2 donde: A es el número de aristas del grafo y N es el número de nodos.
3. La complejidad ciclomática , V(G), de un grafo de flujo G también se define como:
   V (G) = P + 1 donde: *P* es el número de nodos predicado contenidos en el grafo *G*.

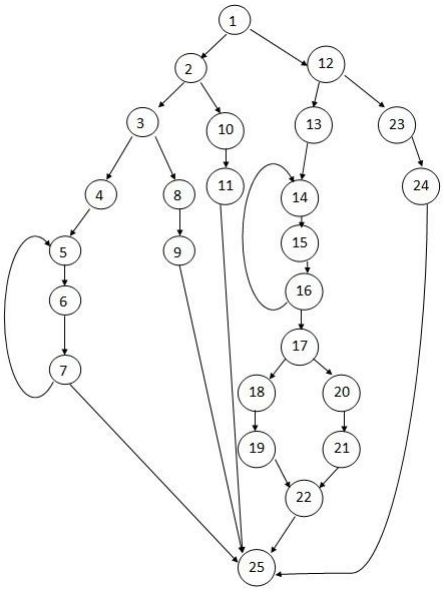


Figura 9 Grafo de Flujo Atender Petición

Para el cálculo de la complejidad ciclomática del Grafo de Flujo Atender Petición fue utilizada la formula V (G)=A-N+2, por tanto la complejidad de grafo es 6, demostrando la cantidad de caminos independientes y por consiguiente la cantidad de pruebas que se deben aplicar para asegurar que se ejecute cada sentencia al menos una vez. Complejidad Ciclomática = 29-25+2=6.

Caminos:

a) 1-2-3-4-5-6-7-25.
b) 1-12-13-14-15-16-17-18-19-22-25
c) 1-2-3-8-9-25.
d) 1-2-10-11.
e) .1-12-13-14-15-16-17-20-21-22-25.
f) 1-12-23-24-25.

Luego de obtenido el Grafo de Flujo y los caminos a recorrer se procederá al diseño de los casos de pruebas que forzaran la ejecución de cada uno de esos caminos. Se escogen los datos de forma que las condiciones de los nodos predicados estén adecuadamente establecidas, con el fin de comprobar cada camino. Para realizar los casos de pruebas es necesario tener en cuenta los siguientes aspectos.

- Descripción: Se realiza la entrada de datos necesaria para validar que los parámetros obligatorios no pasen nulos al método.
- Entrada: Se exponen los parámetros que son necesarios para realizar el método.
- Resultados Esperados: Se define el resultado esperado que debe devolver el método.

En la tabla 4 y 5 se muestran los resultados de los casos de prueba 1 y 2.

Tabla 4 Caso de Prueba del camino 1

Caso de prueba 1	
Camino a:	1-2-3-4-5-6-7-25.
Descripción:	Este camino se encarga de dado una petición de tipo Almacenamiento, activar un Distribuidor de datos en un nodo disponible del clúster permitiendo atender la petición.
Entradas:	Id:1 Nombre de la Media: El Barco Tipo: Almacenamiento IP Solicitante: 10.34.13.14 Difusión: VoD URL: home/Trabajo/ocio/elbarco.avi El fichero XML Proxy : lleno
Resultados: Se espera la invocación por parte del componente Distribuidor de Datos del método Notificar, función que permite eliminar la petición de la lista de procesando y se liberar el hilo utilizado para atenderla. Los resultados obtenidos son satisfactorios.	

Tabla 5 Caso de Prueba del camino 2

Caso de prueba 2	
Camino a:	1-12-13-14-15-16-17-18-19-22-25
Descripción:	Este camino se encarga de dado una petición de tipo *Streaming*, activar un Colector de Datos en un nodo disponible del clúster permitiendo atender la petición.
Entradas:	Id:1 Nombre de la Media: El Barco Tipo: *Streaming* IP Solicitante: 10.8.107.14 Difusión: VoD El fichero XML Proxy: lleno
Resultados: Se espera la invocación por parte del componente Colector de Datos del método Notificar, función que permite eliminar la petición de la lista de procesando y se liberar el hilo utilizado para atenderla. Los resultados obtenidos son satisfactorios.	

Después de aplicada la fase de pruebas se pudo comprobar que el flujo de trabajo es correcto para las funciones probadas y que cumple con las condiciones planteadas.

4.4.Conclusiones Parciales

Durante este capítulo se llevaron a cabo los flujos de trabajo: Implementación y Prueba. Como resultado de estos flujos se generaron los artefactos pertinentes a cada uno de ellos como el Modelo de Implementación, lográndose traducir el diseño del sistema en términos de componentes ejecutables. El proceso de implementación permitió la obtención de un prototipo funcional del manejador de clúster, que cumple en su totalidad con los objetivos generales de la presente investigación y con las expectativas del cliente. Además, se llevó a cabo la técnica de prueba de caja blanca mediante el método de camino básico, definiendo los casos de pruebas que permitieron valorar los resultados obtenidos evidenciando así la calidad del componente desarrollado.

Conclusiones Generales

Con el desarrollo del presente trabajo se cumplieron las tareas y objetivos propuestos obteniendo un prototipo funcional del componente Manejador de Clúster, para lograrlo se realizaron diferentes actividades:

- Se realizó un estudio de las soluciones existentes, demostrando la falta de información referente a los manejadores de clúster para servidores *streaming*, pero permitió conocer elementos de vital importancia en la comprensión de módulos y conceptos con los que el componente debe trabajar.
- Se analizaron las diferentes herramientas y tecnologías teniendo en cuenta su ajuste al diseño arquitectónico del sistema, que cumplieran las expectativas del cliente y que pudieran ser desplegadas sobre la plataforma GNU/Linux.
- Se desarrollo el análisis y el diseño del componente permitiendo un mejor entendimiento del entorno del sistema, ya que fue definido el Modelo del Dominio. El desarrollo de estos flujos ayudó a precisar qué es lo que el componente debe hacer para satisfacer las necesidades del cliente.
- Se llevo a cabo la implementación del sistema permitiendo obtener un prototipo funcional del componente Manejador de Clúster cumpliendo con el objetivo principal de esta investigación. Además, se llevó a cabo la prueba de caja blanca permitiendo valorar las resultados obtenidos evidenciando la calidad del componente.

Recomendaciones

Luego de haber concluido el sistema propuesto y cumplido los objetivos trazados, se plantean las siguientes recomendaciones:

- Someter el sistema a pruebas de calidad de *software*, ya que las mostradas en este trabajo fueron ejecutadas por el programador del sistema y es características humana natural que el creador de una obra, en este caso un *software*, tiene cierta afinidad por el objeto construido; por lo que psicológicamente el programador no querrá “destruir” su creación y no realizará la fase de prueba con la calidad requerida.
- Continuar el desarrollo de la aplicación llevando a cabo la implementación de las funcionalidades que no fueran resueltas en la presente investigación como la réplica del componente para que sea tolerante a fallos de red.
- Realizar la integración del Manejador de Clúster con los módulos que componen el servidor *streaming* distribuido.
- Realizar la notificación mediante AMI[13] o AMD[14], y no mediante métodos auxiliares.
- Profundizar el estudio del servicio IceGrid ofrecido por el *middleware* ICE, ya que permite entre otras cosas la réplica y el balanceado de carga.

[13] Invocación de métodos asíncrona o AMI (Asynchronous Method Invocation)
[14] Tratamiento de métodos asíncrono o AMD (Asynchronous Method Dispatch)

Bibliografía Referenciada

Águila, C. I. (27 de abril de 2010). Herramientas CASE aplicadas. Universidad Mariano Gálvez de Guatemala, Facultad de Ingeniería en Sistemas de Información, Guatemala.

Alonso, D. V., & IDICT. (2001). Gerencia de los lenguajes documentarios. Bibliociencias.

Balladini, Javier A. 2008. Un sistema de vido bajo demanda a gran escala tolerante a fallos de red. [pdf] Barcelona, España : Univercidad Autónoma de Barcelona, 2008.

Cornejo, José Enrique González. 1988. Document Information Retrieval Systems. [En línea] Edificio Century, Irarrázaval 2821, Oficina A-702, 1988. [Citado el: 3 de 12 de 2010.] http://www.docirs.cl/uml.htm..

Escobar, Christian Iván Mejía. Noviembre 2007. Herramienta autoconfigurable para el desarrollo de. México, D.F. : UNIDAD ZACATENCO,DEPARTAMENTO DE COMPUTACIÓN, Noviembre 2007.

Espinosa, J. Koldobika, y otros. 2006. El Uso de la Tecnología Streaming Multimedia en la Educación Superior On-Line. [pdf] Bilbao : Universidad del País Vasco / Euskal Herriko Unibertsitatea, 2006.

Fernández, David Vallejo. 2006. Documentación de ZeroC ICE. s.l. : UNIVERSIDAD DE CASTILLA-LA MANCHA, 2006.

Jacobson, Ivar; Booch, Grady; Rumbaugh, Jim. 2000. El Lenguaje Unificado de Modelado. Manual de Referencia. s.l. : Pearson Education S.A., 2000.

Iribarne Martínez, Luis F. 2003. Un Modelo de Mediación para el Desarrollo de Software basado en Componentes COTS. 2003.

ICE, ZeroC. 2004. zeroc.com. zeroc.com. [En línea] Zeroc, 2004. [Citado el: 16 de octubre de 2010.] http://www.zeroc.com/ice.html.

Inc, SlideShare. 2008. slideshare. slideshare. [En línea] SlideShare Inc, 2008. [Citado el: 12 de noviembre de 2010.] www.slideshare.net/aavasquezca/software-libre. 2003626.

José Ramón Pérez Agüera, Rodrigo Sánchez Jiménez y Jorge Caldera Serrano. 2004. ADAPTACIÓN DE TECNOLOGÍAS STREAM Y XML A CENTROS DE DOCUMENTACIÓN Y TELEVISIÓN. 2004.

Juanes Méndez, Velasco Marcos, Cabrero Fraile, Sánchez Llorente y Rodríguez Conde, M.J. 2010. Recursos tecnológicos audiovisuales de formación en red: sistemas streaming media y teleinmersivos. [pdf] España : Revista Teoría de la Educación: Educación y Cultura en la Sociedad de la Información., Universidad de Salamanca, 2010. Vol. Vol. 11.

Lacort, Alberto Montañola. 2007. Gestor de contenidos de vídeo bajo demanda. [pdf] Lleida : Universidad de Lleida, 2007.

Lidia Fuentes, José M. Troya y Antonio Vallecillo. 2001. Desarrollo de Software Basado en Componentes. [pdf] Málaga, España : Universidad de Málaga, 2001. 29071.

M. Barreiro, V. M. Gulías, J. Mosquera, J. J. Sánchez. 2001. Utilización de programación funcional distribuida y cluster Linux en el desarrollo de servidores de video bajo demanda. [pdf] A Coruña.España : Universidade da Coruñaa, 2001.

Michi Henning y Mark Spruiell. 2008. Distributed Programming with ICE. [pdf] s.l. : Zeroc, 2008.

Mordred, Sir. 2003. VSantivirus. VSantivirus. [En línea] 25 de mayo de 2003. [Citado el: 2010 de diciembre de 3.] http://www.vsantivirus.com/vul-dss-streaming.htm.

Muñoz, Jesús M. Conesa. 2009. Desarrollo de un sistema para toma de decisiones en situaciones de intrusión en instalaciones e infraestructuras. Madrid : s.n., 2009.

Pérez, José Carlos Cortizo. 2007. CORBA. [pdf] Madrid : D. Sistemas Informáticos de la Universidad Europea de Madrid, 2007.

Pressman. 2002. Ingieneria de software un enfoque práctico. 5ta edición. 2002.

Rivera, Fernando Pujaico. 2008. Lenguaje C++ - NIVEL I. [pdf] s.l. : fpujaico_cidfiee@uni.edu.pe, 2008.

Roland Tusch, Christian Spielvoge, Markus Kröpfl, László Böszörményi. 2003. An Adaptive Distributed Multimedia Streaming Server in Internet Settings. [pdf] Austria : Klagenfurt University, 2003.

Scribd. 2008. Scribd. *Scribd.* [En línea el:22 de mayo de 2008] 2005. [Citado el: 19 de 11 de 2010.] http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

Torres, Angel Laguna. 2009. Propuesta de Servidor Streaming de software libre para la captura y transmisión de video y sonido digital. [pdf] La Habana : Universidad de las Ciencias Informáticas, 2009. TD_2726_09.

Bibliografía Consultada

Badia, José Luís Andrés López y Enrique Fuster. 2006. *Streaming de video y audio.* 2006.

IBM. 2007. IBM. [En línea] IBM, 2007. http://www.ibm.com/software/awdtools/developer/rose/.

Sciara, Daniel Rijo. Diciembre de 2004. *Fundamentos de Video Streaming.* [pdf] Montevideo-Uruguay : Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República, Diciembre de 2004.

Bucefalo. 2008. *Bucefalo.* [En línea] 2008. http://bucefalo.com.mx.

Teleformación. 2010. EVA. *EVA.* [En línea] septiembre de 2010. [Citado el: 21 de febrero de 2011.]http://eva.uci.cu/file.php/102/Curso_2010-2011/Clases/Semana_04/Conferencia_6/Materiales_complementarios/Fase_de_Inicio._Disciplina_de_Modelamiento_del_Negocio.pdf.

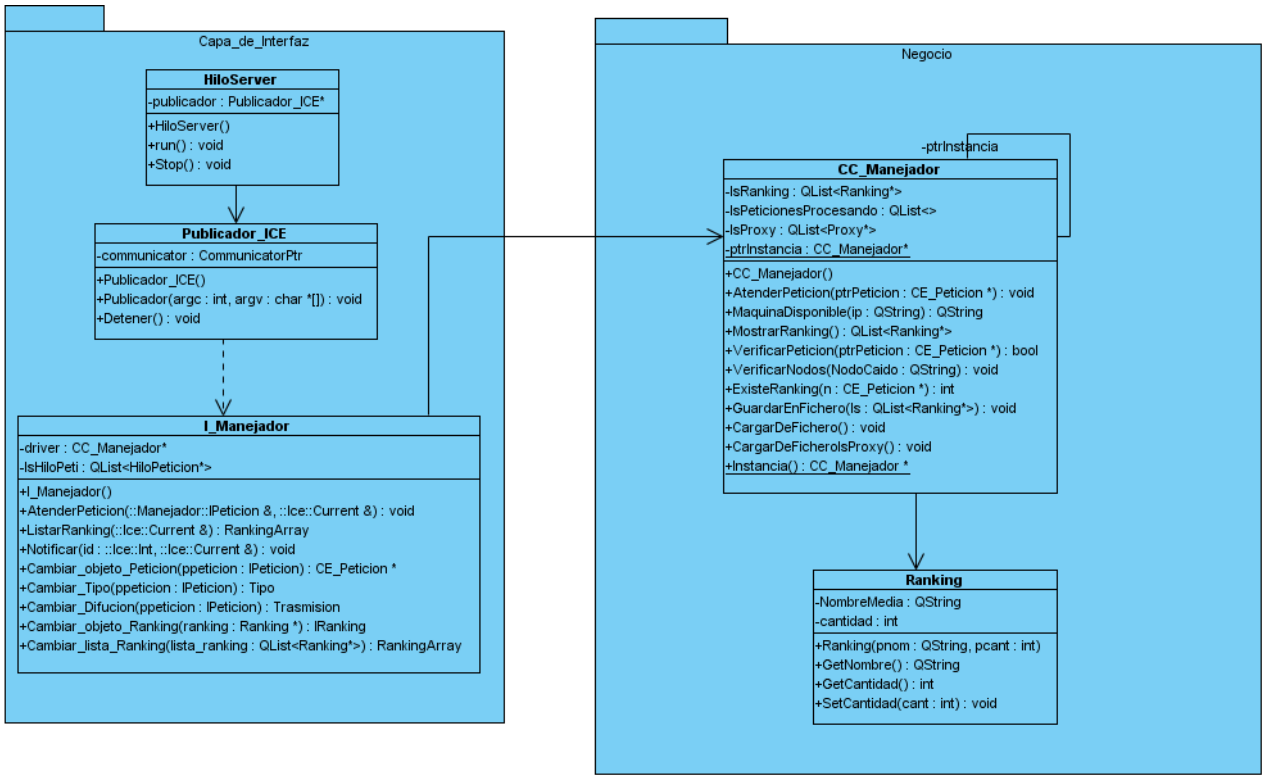
UCI. 2010. *Introducción a la Disciplina de Requisitos de RUP.* [pdf] La Habana : s.n., 2010.

Solingest, Hosting. 2007. Hosting Solingest. *Hosting Solingest.* [En línea] 30 de mayo de 2007. [Citado el: 21 de septiembre de 2010.] http://hosting.solingest.com/cluster-de-servidores.html.

Juanes Méndez, J.A., Velasco Marcos, M.A., Cabrero Fraile, F.J., Sánchez Llorente J.M y Rodríguez Conde, M.J. (2010): Recursos tecnológicos audiovisuales de formación en red: sistemas *streaming* media y teleinmersivos, en Juanes Méndez, J. A. (Coord.) *Avances tecnológicos digitales en metodologías de innovación docente en el campo de las Ciencias de la Salud en España*. Revista Teoría de la Educación: Educación y Cultura en la Sociedad de la Información. Vol. 11, nº 2. Universidad de Salamanca, pp. 214-231 [Fecha de consulta: 30/1/2011].
http://campus.usal.es/~revistas_trabajo/index.php/revistatesi/article/view/7078/7111

ANEXOS

ANEXO1: Diagrama de CD para el caso de uso Mostrar Ranking.



ANEXO2: Caso de Prueba

Caso de prueba 3	
Camino a:	1-2-3-8-9-25.
Descripción:	Este camino se encarga de dado una petición de tipo Almacenamiento, activar un Distribuidor de datos en un nodo disponible del clúster permitiendo atender la petición.

Entradas:	Id:3 Nombre de la Media: El Barco Tipo: Almacenamiento IP Solicitante: 10.34.14.26 Difusión: VoD El fichero XML Proxy : vacío
Resultados: Se espera un mensaje donde se notifica que la lista de proxy está vacía. Los resultados obtenidos son satisfactorios.	